```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#define MAX 100
char st[MAX];
int top = -1;
void push(char st[], char);
char pop(char st[]);
void InfixtoPostfix(char source[], char target[]);
int getpri(char);

void main()
{
   char infix[100], postfix[100];
   printf("\n Enter any infix expression : ");
   gets(infix);
   strcpy(postfix, "");
   InfixtoPostfix(infix, postfix);
   printf("\n The corresponding postfix expression is : ");
   puts(postfix);
}

void InfixtoPostfix(char source[], char target[])
{
   int i = 0, j = 0;
   char temp;
   strcpy(target, "");
   while (source[i] != '\0')
   {
      if (source[i] == '('
```

```
{
   push(st, source[i]);
   i++;
}
else if (source[i] == ')')
{
   while ((top != -1) && (st[top] != '('))
   {
      target[j] = pop(st);
      j++;
   }
   if (top == -1)
   {
      printf("\n INCORRECT EXPRESSION");
      exit(1);
   }
   temp = pop(st);
   i++;
}
else if (isdigit(source[i]) || isalpha(source[i]))
{
   target[j] = source[i];
   j++;
   i++;
}
else if (source[i] == '+' || source[i] == '-' || source[i] == '*' ||
       source[i] == '/' || source[i] == '%' || source[i] == '^')
{
   while ((top != -1) && (st[top] != '(') && (getpri(st[top]) > getpri(source[i])))
   {
      target[j] = pop(st);
```

```
            j++;
          }
          push(st, source[i]);
          i++;
        }
        else
        {
          printf("\n INCORRECT ELEMENT IN EXPRESSION");
          exit(1);
        }
    }
    while ((top != -1) && (st[top] != '('))
    {
        target[j] = pop(st);
        j++;
    }
    target[j] = '\0';
}
int getpri(char op)
{
    if (op == '^')
        return 2;
    else if (op == '/' || op == '*' || op == '%')
        return 1;
    else if (op == '+' || op == '-')
        return 0;
}
void push(char st[], char val)
{
    if (top == MAX - 1)
        printf("\n STACK OVERFLOW");
```

```
    else
    {
        top++;
        st[top] = val;
    }
}
char pop(char st[])
{
    char val = ' ';
    if (top == -1)
        printf("\n STACK UNDERFLOW");
    else
    {
        val = st[top];
        top--;
    }
    return val;
}
```

Output : Enter any infix expression : (a+b/c*(d+e)-f)

The corresponding postfix expression is : abcde+*/f-+